Learning Kernel Logistic Regression in the Presence of Class Label Noise

Jakramate Bootkrajang and Ata Kabán

School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, UK

Abstract

The classical machinery of supervised learning machines relies on a correct set of training labels. Unfortunately, there is no guarantee that all of the labels are correct. Labelling errors are increasingly noticeable in today's classification tasks, as the scale and difficulty of these tasks increases so much that perfect label assignment becomes nearly impossible. Several algorithms have been proposed to alleviate the problem, of which a robust Kernel Fisher Discriminant is a successful example. However, for classification, discriminative models are of primary interest, and rather curiously, the very few existing label-robust discriminative classifiers are limited to linear problems.

In this paper, we build on the widely used and successful kernelising technique to introduce a label-noise robust Kernel Logistic Regression classifier. The main difficulty that we need to bypass is how to determine the model complexity parameters when no trusted validation set is available. We propose to adapt the Multiple Kernel Learning approach for this new purpose, together with a Bayesian regularisation scheme. Empirical results on 13 benchmark data sets and two real-world applications demonstrate the success of our approach.

Key words: Classification, Label noise, Model selection, Multiple Kernel Learning

1 Introduction

Traditional supervised learning machines rely on a correct set of class labels. There is however no guarantee that all the labels will be correct in practice,

Email address: jakramate.b@cmu.ac.th, A.Kaban@cs.bham.ac.uk (Jakramate Bootkrajang and Ata Kabán).

either due to the scale of the labelling task, the lack of information available to determine the class labels or the subjectivity of the labelling experts.

The presence of class label noise inherent in training samples has been reported to deteriorate the performance of the existing classifiers in a broad range of classification problems including biomedical data analysis [20,30] and image classification [24,47]. More recently, class label noise emerges as a side effect of crowdsourcing practices where annotators of different backgrounds are asked to perform labelling tasks. For example Amazon's Mechanical Turk, Citizen science, Galaxy Zoo to name just a few. Although, the problem posed by the presence of class label noise is acknowledged in the literature, it is often naively ignored in practice. Part of the reason for this may be that uniform/symmetric label noise is *relatively* harmless [21,22,12,27].

There is an increasing research literature that aims to address the issues related to learning from samples with noisy class label assignments. The seemingly straightforward approach is by means of data preprocessing where any suspect samples are removed or relabelled [7,1,29,37,31,18]. However, these approaches hold the risk of removing useful data too, which is detrimental to the classification performance, especially when the number of training examples is limited (e.g. in biomedical domains). Most previous approaches try to detect mislabelled instances based on various heuristics, and very few take a principled modelling approach — with the notable exceptions of [32,24,25,36].

Lawrence and Schölkopf [24] incorporated a probabilistic model of random label flipping into their robust Kernel Fisher Discriminant (rKFD) for binary classification. Based on the same model, Li et al. [25] conducted extensive experiments on more complex data sets, which convincingly demonstrated the value of explicit modelling. The rKFD was later extended to multi-class setting by [3] and this has further motivated the recent development of a label noise-tolerant Hidden Markov Model to improve segmentation [15].

While all these works demonstrate the great potential and flexibility of a model based approach, most existing work falls in the category of generative methods. For classification problems, discriminative methods are of interest, and similar algorithmic developments for discriminative classifiers are still limited. For example, Madger et al. [28] studied logistic regression with known label flip probabilities and they reckon problems when these probabilities are unknown. Hausman et al. [17] has given a foundation of a statistical model for the binary classification problem but provide no algorithmic solution to the learning of label noise parameters.

Recently Raykar et al. [36] proposed an EM algorithm to learn a latent variable model extension of logistic regression, for data with multiple sets of noisy labels. Our initial work [4] suggested a more efficient gradient-based algorithm

to optimise a similar latent variable model for problems where only a single set of labels is available. A sparse extension of the model has also been developed in [4]. However all of these developments are limited to linear problems. In this paper we focus on non-linear classification with labelling errors which is not as trivial as it might look at first.

Since the introduction of the kernel trick, many linear classifiers have been harnessed with an ability to solve non-linear problems, whereby their usage extends to a wider range of applications. Generally, deploying a kernel machine also involves determining good kernel parameters, and Cross-Validation (CV) has long been an established standard approach. However, when class label noise is present, it becomes unclear why would CV be a good approach since then all candidate models will be validated against noisy class labels. The issue has also been briefly discussed in [24,6]. In [24], the authors resort to using a 'trusted validation set' to select optimal kernel parameters. The trusted set must be labelled carefully, which seriously restricts the applicability of the method. For example in crowdsourcing it would be very difficult (if not impossible) to construct such a trusted set.

We start by straightforwardly formulating a robust Kernel Logistic Regression (rKLR) as an extension of the robust Logistic Regression (rLR). We present a simple yet effective algorithm to learn the classifier and investigate whether or not CV is a reasonable approach for model selection in the presence of labelling errors. As we shall see, we find that performing CV in noisy environments gives rise to a slightly under-fitted model. We then propose a robust Multiple Kernel Logistic Regression algorithm (rMKLR) based on the so-called Multiple Kernel Learning (MKL) framework (an extensive survey in recent advances of MKL is given in [16]) and the Bayesian regularisation technique [9] to automate the model selection step without using any cross-validation. From this we obtain improvements in both generalisation performance and learning speed. The genealogy of the proposed methods is summarised in Figure 1, which serves as a roadmap for the next section.

Throughout this work, similarly to the related work above, we will focus on label noise occurring at random – that is, the flipping of labels is assumed to be independent of the contents of the data features. The reason for this is simplicity and generic applicability. Alternative models of label noise are discussed after the Experiments section.

2 Robust Kernel Logistic Regression

Consider a set of training samples $\mathcal{D} = \{(\mathbf{x}_n, \tilde{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^m$ and $\tilde{y}_n \in \{0, 1\}$ denotes the observed (possibly noisy) label of $\mathbf{x}_n$. Kernel logistic

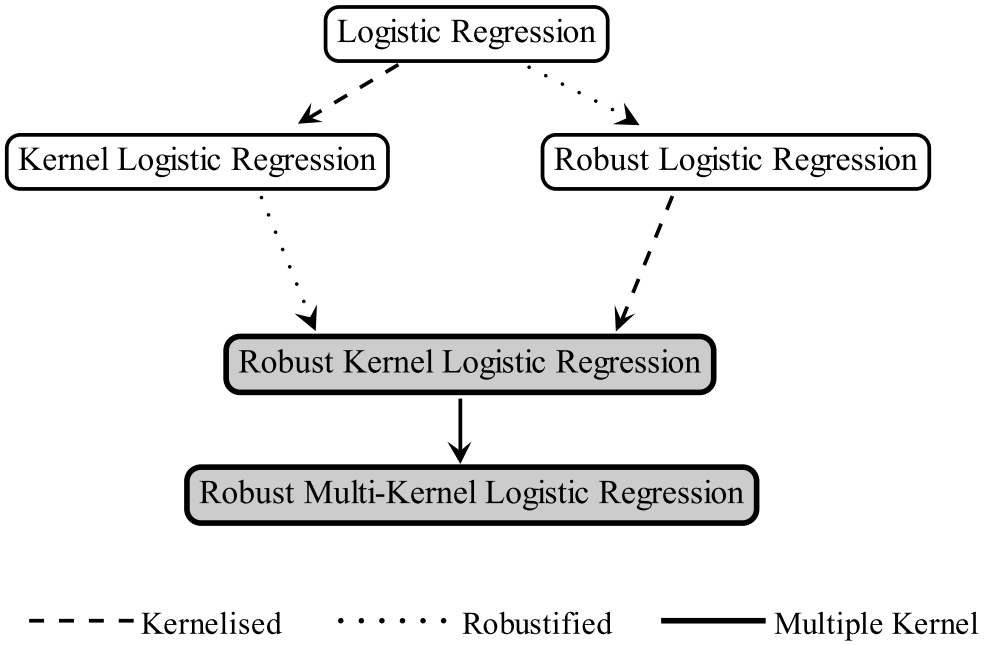

Fig. 1. Genealogy of the robust Kernel Logistic Regression and the robust Multi-Kernel Logistic Regression methods. The highlighted boxes are the classifiers proposed in this paper. Note that there are two paths to arrive at the robust Kernel Logistic Regression.

regression produces a nonlinear decision boundary, $f(\mathbf{x})$, by forming a linear decision boundary in the space of the non-linearly transformed input vectors. By the representer theorem [19], the optimal $f(\mathbf{x})$ has the form:

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\cdot, \mathbf{x}_n) \tag{1}$$

where $\kappa(\cdot, \cdot)$ is a positive definite reproducing kernel that gives an inner product in the transformed space.

Denoting by $\mathbf{w}$ the parameter vector with entries $w_n, n = 1, ..., N$, we define the probability of an observed label $\tilde{y}_n$ as a linear combination of the probabilities that the true label of a point is 0 or 1 respectively:

$$\begin{aligned} p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) &= \sum_{j=0}^{1} p(\tilde{y} = k|y = j) p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) \\ &= \sum_{j=0}^{1} \omega_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}) \end{aligned} \tag{2}$$

Here, $p(\tilde{y} = k|y = j) = \omega_{jk}$ are probabilistic factors representing the probability that the true label j flips into the observed label k. These parameters form a label transition table, Ω, that we will refer to as the *flip matrix*. The full set of parameters for this robust model will be denoted as $\Theta = \{\mathbf{w}, \Omega\}$.

Now, fitting the robust kernel logistic regression is equivalent to maximising the following log-likelihood:

$$\mathcal{L}(\Theta) = \sum_{n=1}^{N} \sum_{k=0}^{1} \mathbb{1}(\tilde{y}_n = k) \log p(\tilde{y}_n = k | \kappa(\cdot, \mathbf{x}_n), \Theta) - \zeta \sum_{n=1}^{N} w_n^2 \tag{3}$$

where $\mathbb{1}(\cdot)$ is the Kronecker delta function. We also included an $L2$ regularisation term to express our preference for a smooth (and non-sparse) model.

In eq. (3), the term $p(\tilde{y}_n = k | \kappa(\cdot, \mathbf{x}_n), \Theta)$ is defined in eq. (2), in which we use a sigmoid function to model the probability of the true label:

$$p(y = 1 | \kappa(\cdot, \mathbf{x}_n), \mathbf{w}) = \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) = \frac{1}{1 + \exp(-\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))} \tag{4}$$

Learning the robust model requires us to estimate the weight vector $\mathbf{w}$ as well as the label-flipping probabilities ω_{jk}. To optimise the weight vector, we can use any non-linear optimiser. We decided to employ conjugate gradients because of its well known computational efficiency, which basically performs the Newton update step along the direction $\mathbf{u} = \mathbf{g} - \mathbf{u}^{old}\nu$. Here $\mathbf{g}$ is the gradient of the log-likelihood w.r.t the weight vector.

Define $\tilde{P}_n^k = p(\tilde{y} = k | \kappa(\cdot, \mathbf{x}_n), \Theta)$, the gradient is given by:

$$\begin{aligned} \mathbf{g} = \sum_{n=1}^{N} \Bigg[\Bigg(& \frac{\mathbb{1}(\tilde{y}_n = 1)(\omega_{11} - \omega_{01})}{\tilde{P}_n^1} + \frac{\mathbb{1}(\tilde{y}_n = 0)(\omega_{10} - \omega_{00})}{\tilde{P}_n^0} \Bigg) \\ & \times \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))(1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n))) \times \kappa(\cdot, \mathbf{x}_n) \Bigg] - 2\zeta \sum_{n=1}^{N} w_n \end{aligned} \tag{5}$$

The Hestenes-Stiefel formula, $\nu = \mathbf{g}^T(\mathbf{g} - \mathbf{g}^{old}) / (\mathbf{u}^{old})^T(\mathbf{g} - \mathbf{g}^{old})$ is used to calculate the step length. The update equation for $\mathbf{w}$ is then the following:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \frac{\mathbf{g}^T \mathbf{u}}{\mathbf{u}^T \mathbf{H} \mathbf{u}} \mathbf{u}, \tag{6}$$

where the Hessian matrix $\mathbf{H}$ is calculated only once at the first iteration. We should note that other schemes such as the Flectcher-Reeves or Polak-Ribère formulae could also be used.

The following multiplicative update equations are then used to update the elements of the flip matrix. These are derived as in [3] using the method of

Lagrangian multipliers to ensure that probabilities sum to 1.

$$\omega_{00} = \frac{\omega_{00} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\omega_{00} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \omega_{01} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} (1 - \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \tag{7}$$

$$\omega_{11} = \frac{\omega_{11} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]}{\omega_{10} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=0)}{\tilde{P}_n^0} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right] + \omega_{11} \sum_{n=1}^{N} \left[\frac{\mathbb{1}(\tilde{y}_n=1)}{\tilde{P}_n^1} \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_n)) \right]} \tag{8}$$

With all the ingredients in place, the learning algorithm is then to alternate between updating each parameter in turn, until convergence. Given an unseen query point $\mathbf{x}_q$, we predict that $\hat{y}_q = 1$ whenever $p(\hat{y} = 1 | \kappa(\cdot, \mathbf{x}_q), \mathbf{w}) = \sigma(\mathbf{w}^T \kappa(\cdot, \mathbf{x}_q))$ returns a value greater than 0.5, and $\hat{y} = 0$ otherwise. This algorithm to efficiently learn rKLR is summarised below in Algorithm 1.

Algorithm 1 Optimisation of rKLR

Input: κ, Ω
  Initialise $\mathbf{w} \leftarrow 0$
  while Iteration < MaxIteration **do**
    Update $\mathbf{w}$ using eq.(5)
    Update Ω using eq.(7) and eq.(8)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Ω.

2.1 Connection to EM based optimisation

As an alternative to the above approach, the algorithm developed in [36] in the context of multiple sets of noisy labels could also be instantiated for our problem. The method in [36], developed in the data space (or equivalently linear kernel case), proposes an EM algorithm to learn a similar model where the true label is modelled as a hidden variable. Instead, we had these hidden variable integrated out when optimising the parameters. It is hence interesting to see how these two algorithms compare.

Similar to [36], let y_n be the hidden true labels, and denote $P_n := p(y_n = 1 | \mathbf{x}, \mathbf{w}, \tilde{y}_n)$ the posteriors of these. To make the link, the expected complete log likelihood (so-called Q-function) in the data space can then be written as:

$$\mathcal{Q}(\Theta) = \sum_{n=1}^{N} P_n \log(\omega_{11}^{\tilde{y}_n} \omega_{10}^{1-\tilde{y}_n} \sigma(\mathbf{w}^T \mathbf{x}_n)) + (1 - P_n) \log(\omega_{01}^{\tilde{y}_n} \omega_{00}^{1-\tilde{y}_n} (1 - \sigma(\mathbf{w}^T \mathbf{x}_n))) \tag{9}$$

- The E-step involves updating P_n based on the given data and the current

estimate of Θ:

$$P_n = \frac{\omega_{11}^{\tilde{y}_n}\omega_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n)}{\omega_{11}^{\tilde{y}_n}\omega_{10}^{1-\tilde{y}_n}\sigma(\mathbf{w}^T\mathbf{x}_n) + \omega_{01}^{\tilde{y}_n}\omega_{00}^{1-\tilde{y}_n}(1-\sigma(\mathbf{w}^T\mathbf{x}_n))} \tag{10}$$

- The M-step then re-estimates the parameters using P_n from the E-step. For example the gradient for updating the weight vector is given by:

$$\mathbf{g} = \sum_{n=1}^{N}\Big(P_n - \sigma(\mathbf{w}^T\mathbf{x}_n)\Big)\mathbf{x}_n \tag{11}$$

  Likewise, ω_{11} is updated using:

$$\omega_{11} = \frac{\sum_{n=1}^{N} P_n\tilde{y}_n}{\sum_{n=1}^{N} P_n} \tag{12}$$

Now observe that substituting eq.(10) into eq.(12), we recover our multiplicative form of updates for ω_{11} — with one subtle but important difference: In the EM approach, P_n used in eq.(12) is computed with the old values of the parameters $\mathbf{w}$. Instead, our multiplicative updates use the latest fresh values of all the parameters they depend on. This not only implies that our algorithm saves the storage cost of the posteriors P_n during the iterations, but it also has a better chance to converge in fewer iterations. The latter can be seen by noting that our algorithm is equivalent to a component-wise EM [10], that is an EM in which each component in the parameter space $\Theta = \{\mathbf{w}, \Omega\}$ is updated sequentially. Component-wise EM has indeed been observed empirically to converge faster than standard EM [10]. We should of course note also that P_n can be useful for interpretation and our algorithm does not compute this explicitly during its iterations. However we can compute P_n after convergence using the final values of the parameters.

2.2 Selecting the kernel width: A multi-kernel approach

For any kernel machine, the value of the kernel parameters are critical to the generalisation performance, and determining these is an important part of the task. Here we focus on radial kernels for the sake of concreteness. In this case the kernel parameter is the width of the kernel. A usual way of finding optimal kernel width is by means of cross-validation (CV). However, the success of this technique relies on the implicit assumption that the test set follows the same distribution as the training set – which is violated in the label noise scenario. One might instead attempt to cross-validate on the out-of-sample model likelihood, but unfortunately this yields unsatisfactory results in our experience as it tends to select a too small value for the kernel width, leading to overfitting. Hence a more substantial modification would be

required CV work, e.g. by finding a way to express the label-robust objective in terms of some regularised objective as it has been done for SVM under *feature noise* [43]. However, this is likely to involve a regularisation parameter to be set as well, and one would then need to perform CV over a 2D grid – a rather resource-inefficient option.

Instead, we propose to employ the Multiple Kernel Learning (MKL) framework, giving it a new purpose. In MKL, a combination of several kernels is learnt in order to get a good representation of the data. We adopt the framework as a method to find optimal kernel width automatically without performing cross-validation. In contrast to the majority of MKL literature where the aim is centred around combining heterogeneous data sources [34,23], our adoption of MKL focuses on the combination of multiple kernels that correspond to different notions of similarity, as defined by different kernel widths. This approach will bypass the need for cross-validation and as a side-effect of this it also speeds up the learning process.

There are several ways to combine kernels. We will use a conic combination, as the following:

$$\kappa(\cdot,\cdot) = \sum_{i=1}^{S} \eta_i \kappa_i(\cdot,\cdot) : \qquad \eta_i \geq 0 : \forall i \tag{13}$$

Conic combinations represent a popular way to combine kernels. It is less constrained than a convex combination would be, and the positivity constraint ensures that the kernel weighting parameters do not cancel out each other. The latter is important since linear combinations may lead to unstable learning [14]. A convex combination could also be used but it would require the extra constraint that η_i sums to unity, which is unnecessary.

In contrast to the case where one is concerned with heterogeneous data sources, we want $\boldsymbol{\eta}$ to be sparse for our purpose, in order to select only a few of a set of possible kernel widths. To implement this idea we use a generalised LASSO-like approach, positing independent exponential priors to enforce this preference on $\boldsymbol{\eta}$. This results in adding a new regulariser to the objective in eq.(3) to accommodate the MKL framework:

$$\sum_{n=1}^{N} \mathbb{1}(\tilde{y}_n = 1) \log \tilde{P}_n^1 + \mathbb{1}(\tilde{y}_n = 0) \log \tilde{P}_n^0 - \zeta \sum_{i=1}^{N} w_i^2 - \sum_{i=1}^{S} \xi_i \eta_i \tag{14}$$

To ensure positivity, we reparametrise $\eta_i = u_i^2$, and optimise for u_i using conjugate gradients method. The derivative of the objective, eq.(14), w.r.t u_i

is given by:

$$\sum_{n=1}^{N}\Bigg[\left(\frac{\mathbb{1}(\tilde{y}_n=1)(\omega_{11}-\omega_{01})}{\tilde{P}_n^1}+\frac{\mathbb{1}(\tilde{y}_n=0)(\omega_{10}-\omega_{00})}{\tilde{P}_n^0}\right)$$
$$\times\,\sigma(\mathbf{w}^T\kappa(\cdot,\mathbf{x}_n))(1-\sigma(\mathbf{w}^T\kappa(\cdot,\mathbf{x}_n)))\times(\mathbf{w}^T\kappa_i(\cdot,\mathbf{x}_n))\Bigg]-2\xi_i u_i \qquad (15)$$

We later recover η_i by squaring the optimised u_i.

2.3 Choosing the regularisation parameters by Bayesian regularisation

As discussed earlier, the use of cross-validation is not straightforward since a naive use is questionable in the presence of labelling errors. This includes the selection of the regularisation hyper-parameters. To circumvent the problem, we adopt a Bayesian regularisation technique to automatically determine good values of ζ and $\boldsymbol{\xi} := (\xi_1, ..., \xi_S)$. For this, we consider a Bayesian interpretation of eq.(14).

Consider the terms that depend on the parameter $\mathbf{w}$ and ζ first. The posterior probability of $\mathbf{w}$ can be expressed as:

$$p(\mathbf{w}|\mathcal{D},\zeta)\propto p(\mathcal{D}|\mathbf{w},\boldsymbol{\xi})p(\mathbf{w}|\zeta) \qquad (16)$$

The first term on the r.h.s corresponds to the data likelihood while the second term is our regularisation term for $\mathbf{w}$. By taking logarithm on both sides of eq. (16), $\log p(\mathbf{w}|\mathcal{D},\zeta)=\log p(\mathcal{D}|\mathbf{w},\boldsymbol{\xi})+\log p(\mathbf{w}|\zeta)+const.$, we see that the regularisation term is simply the negative logarithm of the prior distribution conditioned on ζ, the regularisation parameter. Therefore, $p(\mathbf{w}|\zeta)=\mathcal{N}(0,1/\zeta)$.

We want to eliminate ζ from the formulation, so we build the model further by putting a prior on ζ. We choose this to be an exponential distribution because the values of ζ must be positive: $p(\zeta|\beta)=\beta e^{-\beta\zeta}$. Here, β is a hyper-parameter, i.e. the inverse scale of the exponential. This encodes our uncertainty about ζ, and as such, it reflects our uncertainty about $\mathbf{w}$ at a higher level of inference. We used $\beta=2$ in in the reported experiments to constrain the expected prior variance of $\mathbf{w}$.

With this hyper-prior in place, we can write the marginal prior distribution, $p(\mathbf{w})$ by integrating out ζ:

$$p(\mathbf{w})=\int_0^\infty p(\mathbf{w}|\zeta)p(\zeta)\mathtt{d}\zeta \qquad (17)$$

Completing the integration by the use of the Gamma integral $\int_0^\infty x^{\nu-1}e^{-\mu x}\mathtt{d}x=$

$\frac{\Gamma(\nu)}{\mu^\nu}$, we obtain:

$$\begin{aligned} p(\mathbf{w}) &= \int_0^\infty \prod_{i=1}^m \left\{ \sqrt{\frac{\zeta}{2\pi}} e^{-\frac{\zeta}{2} w_i^2} \right\} \cdot \beta e^{-\beta\zeta} \mathtt{d}\zeta \\ &= \frac{\beta}{(2\pi)^{m/2}} \int_0^\infty \zeta^{(m/2+1)-1} e^{-\zeta(\frac{1}{2}\sum_{i=1}^m w_i^2 + \beta)} \mathtt{d}\zeta \\ &= \frac{\beta}{(2\pi)^{m/2}} \frac{\Gamma(\frac{m}{2}+1)}{(\frac{1}{2}\sum_{i=1}^m w_i^2 + \beta)^{(m/2+1)}} \end{aligned} \tag{18}$$

Going back to our objective function in eq. (14), we now replace $\mathbf{w}$'s the regularisation term with the negative log of the newly derived marginal prior, and optimise this objective w.r.t. $\mathbf{w}$. Computing the gradient of this new regularisation term yields:

$$-\frac{\partial \log p(\mathbf{w})}{\partial \mathbf{w}} = \frac{\frac{m}{2}+1}{\frac{1}{2}\sum_{i=1}^m w_i^2 + \beta} \frac{\partial \sum_{i=1}^m w_i^2}{\partial \mathbf{w}} \tag{19}$$

and since this has the same form as the gradient of the original regularisation term would, we read off from eq. (19) the regularisation parameter as,

$$\zeta = \frac{\frac{m}{2}+1}{\frac{1}{2}\sum_{i=1}^m w_i^2 + \beta} \tag{20}$$

Next we proceed to treat ξ_i using the same technique of Bayesian regularisation. This time we are looking for ξ_i that produces a sparse $\boldsymbol{\eta}$, in order to select just a very few kernel widths. We will employ a regularisation on each component of $\boldsymbol{\eta}$. The Bayesian interpretation of eq.(14) with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ is given by,

$$p(\boldsymbol{\eta}|\mathcal{D}, \mathbf{w}, \boldsymbol{\xi}) \propto p(\mathcal{D}|\boldsymbol{\eta}) \prod_{i=1}^S p(\eta_i|\xi_i) \tag{21}$$

where we employed independent priors distributions on each η_i. Recall that we constrained η_i to be non-negative, so a natural choice is to use independent exponential distributions $p(\eta_i|\xi_i) = \xi_i e^{-\xi_i \eta_i}$, and ξ_i denote the inverse scale parameters of these. These are hyperparameters that correspond to the regularisation parameters in the last term of our objective function eq. (14).

Again, we want to integrate out the ξ_i from the formulation, so we build this model further, positing a hyper-prior on all ξ_i. These also need to be positive, hence we use the exponential distribution one more, $p(\xi_i) = \psi e^{-\psi \xi_i}$, and set $\psi = 10^{-100}$ to a non-informative hyperprior that will encourage a sparse solution.

We obtain the marginal prior by integration, which gives:

$$p(\eta_i) = \int_0^\infty \xi_i e^{-\xi_i \eta_i} \cdot \psi e^{-\psi \xi_i} \mathtt{d}\xi_i$$
$$= \psi \int_0^\infty \xi_i^{(1+1)-1} e^{-\xi_i(\eta_i+\psi)} \mathtt{d}\xi_i = \psi \frac{\Gamma(2)}{(\eta_i+\psi)^2} \tag{22}$$

Finally, replacing the negative log of this in place of our original regularisation term in eq. (14), re-parametrising $\eta_i^2 = u_i$ and taking derivative of the log of eq.(22) w.r.t u_i we have,

$$-\frac{\partial \log p(\eta_i)}{\partial u_i} = \frac{2}{(\eta_i+\psi)} \frac{\partial \eta_i}{\partial u_i} \tag{23}$$

From here we read off that

$$\xi_i = \frac{2}{\eta_i + \psi} \tag{24}$$

Algorithm 2 summarises the steps to learn our novel "robust Multiple Kernel Logistic Regression" (rMKLR) model.

Algorithm 2 Optimisation of rMKLR

Input: Set of predefined kernels $\kappa_{i=1:S}$, Ω
  Initialise $\mathbf{w} \leftarrow 0$, $\boldsymbol{\eta} \leftarrow 1$, $\zeta \leftarrow 0$, $\boldsymbol{\xi} \leftarrow 0$
  while Iteration < MaxIteration **do**
    Update $\mathbf{w}$ using eq.(5)
    Update ζ using eq.(20)
    Update η_i by optimising u_i using eq.(15) and set $\eta_i = u_i^2$
    Update ξ using eq.(24)
    Update Ω using eq.(7) and eq.(8)
  end while
Output: Optimised weight vector, $\mathbf{w}$. Optimised Ω.

3 Experiments

We conducted extensive experiments to answer three main research questions.

- Firstly, we ask if rKLR improves KLR in terms of robustness against labelling errors as measured via classification performance. To answer this question, we also study the relative harm of two common types of label noise: symmetric and asymmetric noise. Symmetric noise is when the same percentage of class labels flip from one class into the other, while asymmetric noise is when labels from one class flip into the other but not vise-versa.

- Secondly, we ask if MKL can be used to find a suitable kernel parameter in noisy settings. To answer the second research question we first show that our proposed MKL for kernel width selection works in a noise-free scenario. We then progress to show the comparative performance of rKLR where its kernel width was selected using 1) CV with a trusted validation set, 2) CV without a trusted validation set and 3) MKL framework in a noisy setup.
- Thirdly, we ask how the proposed rMKLR compares to robust Kernel Fisher Discriminant (rKFD) [24], to the gold-standard Support Vector Machine (SVM) and to the Stochastic Programming for Multiple Kernel Learning (StPMKL) [46]. The recently proposed StPMKL is designed to learn from noisy labels by relaxing a deterministic constraint in MKL into a chance constraint using a binary random variable for each example that indicates if the class assignment of the example is correct. It has been shown to outperform state-of-the-art MKL algorithms in noisy setups [1], and hence can be regarded as one of the best MKL algorithms for noisy labels. SVM is an established classifier which incorporates slack variables, hence it should be robust to label noise to some extent [2]. Comparison with *vanilla* SVM will also reveal to what extent one could regard label noise as part of a the traditional classification problem, and whether or not the extra effort of noise modelling is actually needed. The rKFD was included in this comparison because it has been previously found to be effective in a wide range of noisy non-linear problems [25]. It is a generative classifier as opposed to our rMKLR – which is a discriminative classifier – and it is interesting to see their performance comparatively.

3.1 Experimental protocol

In answering these questions, we train the proposed classifier on data where label noise is created artificially so as to gain better understanding of its effects. We train the model on the corrupted data set and evaluate the learnt model using clean test sets. We consider label noise contamination ranging from 10% up to 40%. We should note that label noise over 40% is very unlikely to occur in practice as it would mean a very poor labelling close to random class assignment.

We use 13 UCI benchmark data sets [35] in our controlled experiments. Each problem has been split into 100 train/test realisations except the *Image* and *Splice* data sets where 20 realisations are provided. The characteristics of the data sets used are summarised in Table 1. We later use crowdsourcing and

[1] StPMKL was compared to Simple MKL [45] and MKL formulation of the robust SVM [44]
[2] We use LIBSVM [11] in our reported experiments

Data set	Training samples	Test samples	Pos. samples	Neg. samples	Dimensionality
Banana	400	4900	44.83%	55.17%	2
B.Cancer	200	77	29.28%	70.72%	9
Diabetes	468	300	34.90%	65.10%	8
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Image	1300	1010	56.95%	43.05%	18
Ringnorm	400	7000	49.51%	50.49%	20
S.Flare	666	400	65.28%	34.72%	9
Splice	1000	2175	44.93%	55.07%	60
Thyroid	140	75	30.23%	69.77%	5
Titanic	150	2051	58.33%	41.67%	3
Twonorm	400	7000	50.04%	49.96%	20
Waveform	400	4600	32.94%	67.06%	21

Table 1
Characteristics of non-linear benchmark data sets.

cheaply annotated data sets to demonstrate real applications of the algorithm in learning from unreliable data sources.

For experiments where cross-validation (CV) is needed, we adopt the 'best practice' described in [8], where model selection is seen as part of the learning and 5-folds CV is performed independently on each split of the data. When needed, we set aside 10% of the training data as a trusted validation set, in which all labels are perfect. We employed a Gaussian Radial Basis Function (RBF) kernel defined as,

$$\kappa(\mathbf{x}, \mathbf{x}_n) = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_n||^2}{\gamma}\right) \tag{25}$$

in all of our experiments except in the textual entailment recognition task where we used a linear kernel.

For MKL, our multiple kernels set is composed of 21 RBF base kernels with widths γ in the set $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$. This set has a comprehensive coverage of the range of possible values and we found this level of granularity works well in practice. An assessment of the sensitivity to this choice will be made in a later section. We also use this set of parameter values in the CV experiments, for searches for both the kernel width and the C parameter in the case of SVM.

3.2.1 Symmetric versus Asymmetric noise

We start with an illustrative experiment, in which we are interested in finding out which kind of random noise is more detrimental to the traditional kernel learning. We train traditional KLR on data with 30% symmetric and 30% asymmetric noise and present the average classification errors and standard deviations over 100 repetitions in Figure 2.

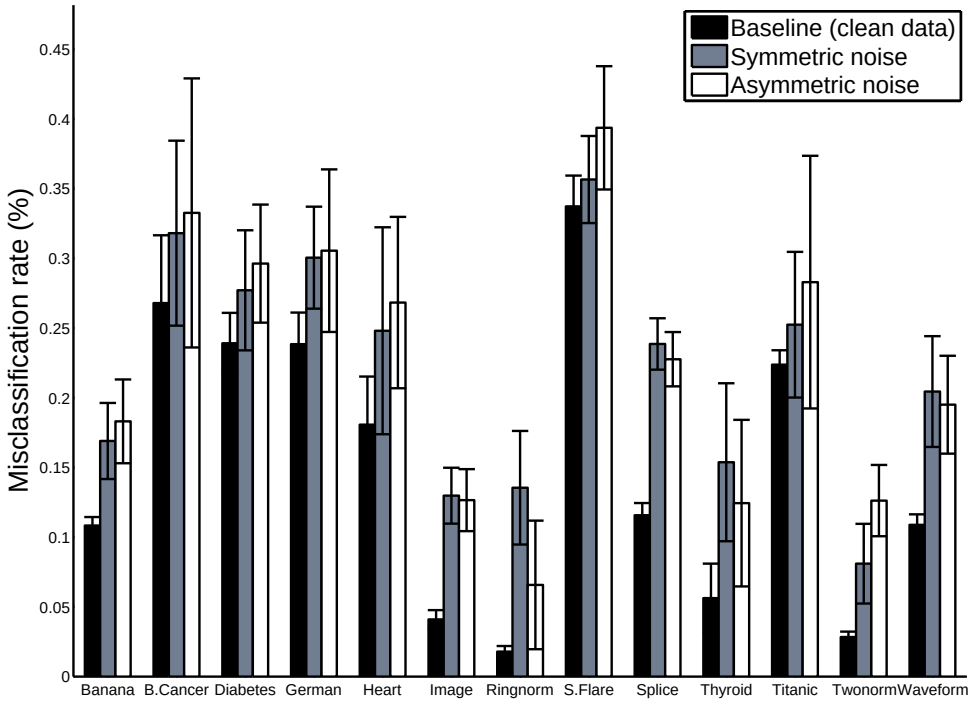

Fig. 2. Effect of 30% symmetric and asymmetric noise to traditional KLR, compared against clean baseline.

Taking the performance on clean data to be the baseline result, we see that in 5 data sets symmetric label noise is more detrimental while for the rest of the data sets asymmetric noise perturbs the classifier more. Hence, as we see, even symmetric label noise is not always harmless. Worth noting also that the effect of label noise is more significant in artificial data that has a low Bayes error (e.g., Ringnorm and Twonorm) than a real world data set (e.g., S.Flare) that might possibly already have some inherent label noise.

3.2.2 The advantage of modelling the label noise

Having seen that traditional KLR is not suitable in learning from data with noisy labels, we are now interested to see if incorporating a label noise model helps improving the classification performance. To this end, we compare KLR to the proposed rKLR. To eliminate any other factors which could affect the assessment, we defer the use of MKL with Bayesian regularisation (hereinafter referred to as "the MKL") until the next subsection, and instead here we use CV with a trusted validation set containing correct labels to select the kernel width parameter for both algorithms. Since we have seen that both types of label noise are detrimental to the classifiers, we performed this test for both types of noise, and pulled together the misclassification rates computed from 100 independent splits of the data contaminated with symmetric noise as well as the 100 splits of data contaminated with asymmetric noise. Table 2 summarises the average misclassification rates, standard deviations and p-values.

We see that rKLR substantially improves upon KLR in the high noise conditions (40%): In 11 out of the 13 data sets tested, we obtain statistically significant improvements as tested using the Wilcoxson ranksum test at the 5% level. Though, interestingly, KLR is quite robust in the low noise (10%) case. We also observe that, as the degree of mislabelling becomes more severe, the performance gaps are getting larger. This can be readily seen in the 30%-40% noise settings. On the basis of these results in Table 2 we may conclude that label noise modelling indeed advantageous and makes a great difference.

3.3 Results: Cross-Validation versus MKL with Bayesian regularisation

3.3.1 Results on clean data

In the literature MKL has been used to combine different sources of data. However, here we adopted the MKL framework to automatically determine a good kernel width. Before proceeding to our noisy scenarios, we will first establish that the MKL works on clean data. We present in Table 3 the comparative classification results between CV and the MKL on the clean benchmark data sets.

From Table 3, we see that the MKL is effective in choosing a good representation of the data, i.e. a good kernel width. We see that the difference between CV and Bayesian regularisation – albeit statistically significant in favour of the latter in 6 out of 13 cases – is small (mostly within $<1\%$) in all cases. This clearly confirms that the MKL with Bayesian regularisation is a both effective and efficient way to automate the process of kernel width selection.

Dataset	Noise level					
	10%			20%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	**12.28 ± 1.49**	12.80 ± 2.10	0.03	14.58 ± 1.96	**12.91 ± 2.32**	$3.21e-22$
B.Cancer	**28.96 ± 5.62**	31.02 ± 6.20	$6.51e-4$	**30.19 ± 7.04**	32.16 ± 7.82	0.02
Diabetes	**24.86 ± 3.09**	26.00 ± 3.48	$3.05e-4$	26.17 ± 3.17	26.69 ± 4.28	0.79
German	**25.07 ± 3.23**	26.90 ± 3.46	$7.62e-8$	**26.64 ± 3.58**	27.78 ± 3.66	$3.27e-3$
Heart	19.84 ± 6.57	20.34 ± 5.73	0.17	22.31 ± 5.10	**20.63 ± 4.76**	$9.91e-4$
Image	6.58 ± 1.13	6.19 ± 1.52	0.12	8.06 ± 1.12	**6.80 ± 1.00**	$2.79e-6$
Ringnorm	4.51 ± 2.23	**3.11 ± 1.78**	$5.91e-13$	4.94 ± 3.23	**3.29 ± 1.86**	$1.87e-6$
S.Flare	34.45 ± 2.33	34.81 ± 2.83	0.24	35.87 ± 2.70	36.00 ± 2.92	0.96
Splice	14.90 ± 1.47	14.90 ± 1.69	0.89	17.33 ± 1.67	16.82 ± 1.63	0.14
Thyroid	9.25 ± 4.00	**7.76 ± 4.28**	$6.194e-5$	9.56 ± 4.55	**8.49 ± 4.60**	$9.86e-3$
Titanic	22.85 ± 1.33	22.88 ± 1.90	0.63	23.57 ± 2.55	23.30 ± 2.39	0.16
Twonorm	4.69 ± 1.16	**3.79 ± 0.78**	$1.51e-19$	7.82 ± 1.88	**4.38 ± 1.20**	$5.33e-54$
Waveform	12.91 ± 1.52	**12.21 ± 1.15**	$2.22e-6$	15.04 ± 2.18	**12.81 ± 1.54**	$1.05e-33$
Dataset	Noise level					
	30%			40%		
	KLR	rKLR	p-value	KLR	rKLR	p-value
Banana	17.60 ± 2.95	**16.13 ± 3.99**	$7.38e-7$	25.63 ± 6.01	**23.08 ± 10.49**	$1.85e-5$
B.Cancer	32.54 ± 8.29	32.92 ± 9.26	0.87	36.89 ± 10.39	35.52 ± 10.36	0.15
Diabetes	28.67 ± 4.37	**27.42 ± 4.56**	$2.60e-4$	33.77 ± 6.19	**31.14 ± 7.03**	$7.70e-6$
German	30.30 ± 4.86	**28.81 ± 4.69**	$9.78e-4$	33.86 ± 8.42	**30.11 ± 4.69**	$2.19e-4$
Heart	25.82 ± 6.87	26.64 ± 8.15	0.62	34.99 ± 8.76	**30.98 ± 11.65**	$2.34e-5$
Image	12.82 ± 2.10	**10.45 ± 3.21**	$1.20e-3$	20.29 ± 3.78	**15.98 ± 7.24**	$8.48e-3$
Ringnorm	10.06 ± 5.57	9.57 ± 6.00	0.43	16.92 ± 8.78	15.78 ± 9.67	0.11
S.Flare	37.51 ± 4.25	36.82 ± 3.63	0.13	41.04 ± 4.71	**38.61 ± 4.37**	$1.50e-7$
Splice	23.31 ± 1.95	**21.20 ± 4.06**	0.03	31.20 ± 3.85	**26.74 ± 8.49**	0.04
Thyroid	13.91 ± 5.99	13.76 ± 8.10	0.24	22.44 ± 11.01	**19.16 ± 13.41**	$8.82e-5$
Titanic	26.77 ± 7.54	**25.19 ± 5.55**	0.03	34.64 ± 12.49	**29.47 ± 11.39**	$4.18e-9$
Twonorm	10.36 ± 3.53	**9.60 ± 6.54**	$1.17e-4$	22.00 ± 6.16	**17.14 ± 13.42**	$6.19e-5$
Waveform	19.97 ± 3.77	**17.37 ± 5.24**	$4.11e-10$	27.50 ± 5.81	**22.86 ± 9.86**	$3.74e-8$

Table 2
The relative performance of KLR and the proposed rKLR. Average errors, standard deviations, and p-values at 5% level.

3.3.2 Results on noisy data

We now move on to more challenging noisy settings. We will now compare the MKL against CV in a scenario where label errors are present. We shall focus on two aspects, firstly how label noise affects CV based model selection, and secondly how does the MKL compare to CV in this setup. We artificially inject 30% random symmetric and asymmetric noise into the training sets, while keeping the test sets clean. We compare the performance of the proposed model in which the kernel widths were selected using: (1) CV on a trusted validation set (2) CV on noisy validation set (3) MKL with Bayesian regularisation.

Data set	Cross validated rKLR	rMKLR	p-value
Banana	10.96 ± 0.81	10.72 ± 0.52	0.06
B.Cancer	29.94 ± 4.65	**27.73 ± 4.19**	$9.17e-4$
Diabetes	24.53 ± 2.21	24.24 ± 1.85	0.47
German	25.38 ± 2.63	**23.52 ± 2.26**	$2.09e-7$
Heart	18.63 ± 4.00	**16.30 ± 3.39**	$7.48e-5$
Image	**3.73 ± 0.71**	5.65 ± 0.96	$2.78e-6$
Ringnorm	1.80 ± 0.43	**1.48 ± 0.10**	$4.16e-12$
S.Flare	**33.50 ± 2.15**	34.33 ± 1.75	$1.05e-3$
Splice	**11.47 ± 0.82**	13.30 ± 1.13	$7.52e-6$
Thyroid	5.96 ± 2.78	5.91 ± 2.70	0.95
Titanic	**22.26 ± 0.94**	22.73 ± 0.83	$4.28e-5$
Twonorm	2.86 ± 0.36	**2.47 ± 0.16**	$4.58e-17$
Waveform	10.78 ± 0.85	**10.58 ± 0.45**	0.03

Table 3
Comparison between standard cross-validation and MKL with Bayesian regularisa-

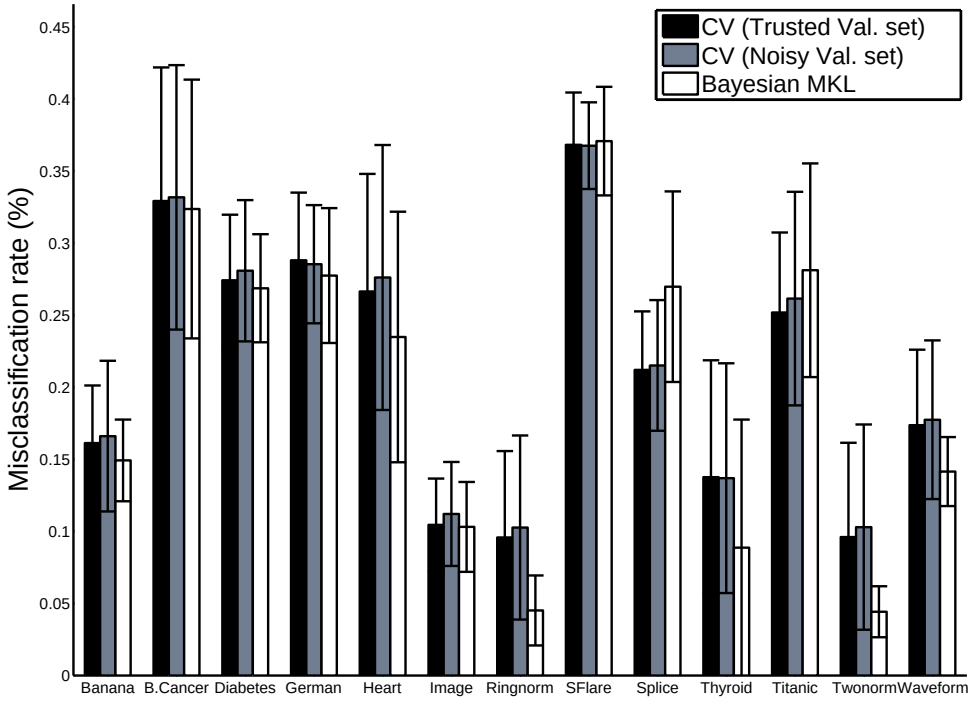

Fig. 3. Cross-validation for kernel width selection on different validation sets versus MKL with Bayesian regularisation.

Figure 3 reports the mean errors and standard deviations from 100 repetitions.

Interestingly, we observe negligible difference between doing CV on the originally noisy validation versus doing CV on a trusted validation set. More surprisingly we notice that CV on noisy data sometimes produces better results than on the trusted validation set – for example in the *Splice* dataset. We conjecture these datasets might originally already contain some label noise.

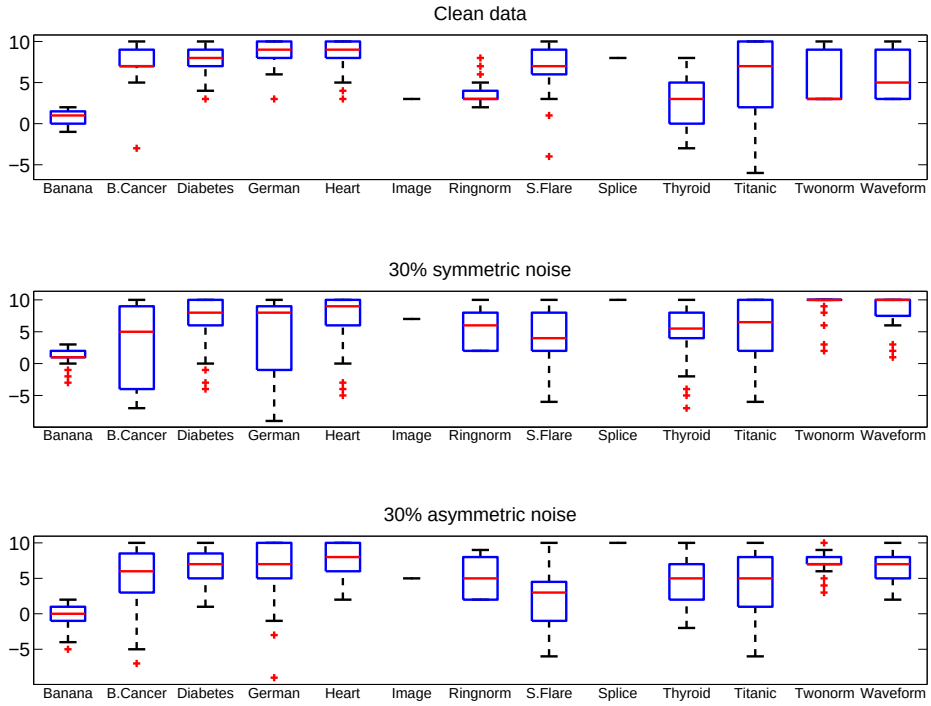

Fig. 4. Comparison of the medians of the kernel widths selected using clean data and two types of label noise at 30% level, averaged over 10 independent runs. Cross–validation was done using noisy validation set

However, the MKL performs better than the others in general. To better understand why CV on noisy validation set is still as good as CV on trusted validation set, we show in Figure 4 plots of kernel widths for each data set from 10 random data splits, where all labels are clean (top), symmetric noises are presented (middle) and asymmetrically noises are presented (bottom). In the noisy scenarios we used noisy validation set to select kernel widths.

Figure 4 reveals that in the case of asymmetric noise the medians of the kernel widths tend to be larger than the ones chosen using clean data while in the symmetric case the widths are mostly in the same proximity as the widths from the clean data. Having a larger width means a wider Gaussian basis function, that is a slight underfitting effect. In the case of rKLR, as tested, it is still legitimate to have a slightly wider width as those points with suspicious labels will likely be flagged as wrong label samples. Consequently CV on noisy labels should not deteriorate classification performance much compared to an idealised CV on trusted validation set – although CV is of course computationally more costly than MKL, as demonstrated in the sequel.

3.3.3 Comparison of the computation time of MKL vs CV

To assess the relative computation time of these methods, we report in Table 4 the average running times for a single training/testing split for each data.

As we can see, MKL with Bayesian regularisation is approximately 5 to 10 times faster than standard CV.

Dataset	CPU time (seconds)		Dataset	CPU time (seconds)	
	rKLR	rMKLR		rKLR	rMKLR
Banana	48.44 ± 6.13	**10.16 ± 0.28**	S.Flare	1033.25 ± 66.55	**26.73 ± 0.74**
B.Cancer	26.61 ± 0.90	**2.96 ± 0.25**	Splice	245.61 ± 1.68	**64.12 ± 1.36**
Diabetes	151.78 ± 44.26	**14.84 ± 0.42**	Thyroid	23.04 ± 1.25	**1.64 ± 0.29**
German	192.84 ± 54.27	**31.63 ± 3.34**	Titanic	15.81 ± 1.74	**1.68 ± 0.12**
Heart	24.23 ± 0.93	**2.15 ± 0.09**	Twonorm	49.19 ± 0.81	**10.75 ± 0.36**
Image	1218.50 ± 451.54	**106.99 ± 1.41**	Waveform	49.69 ± 1.03	**10.07 ± 1.14**
Ringnorm	51.21 ± 0.78	**10.54 ± 0.34**			

Table 4
Running times on a 2.67GHz Intel Core i5 CPU averaged over 10 random splits. The MKL (rMKLR) is 5 to 10 times faster than the traditional CV approach.

3.3.4 Assessing the sensitivity of rMKLR to the number of kernel width choices

Throughout we have used the set of 21 kernel width values to select from in the above experiments. It is then interesting to see how the number of kernels in this set would affect the performance of the proposed technique. To this end, we fix the range of the kernel width choice to the interval $[2^{-10}, 2^{10}]$ as before, but vary the number of available kernel width choices within this range from 11 up to 161. We inject a mix of symmetric and asymmetric noise at 30% level into the training sets and validate the model on clean test sets. The results are given in Figure 5 and show that although the classification performances vary somewhat as the size of the set of kernel width choices varies, the differences in most cases are marginal, with the standard error bars are highly overlapped. This clearly demonstrates that the number of kernels in the kernel width set has a small effect on the classification performance and we are free to choose any reasonable configuration. However one has to bear in mind that having larger kernel set means a longer training time.

3.4 Comparisons with state-of-the-art classifiers and other label-robust approaches

In our final set of controlled experiments, we compare rMKLR to state-of-the-art nonlinear classifiers that have robustness properties: rKFD [24], the SVM, RP-Kernel-Perceptron [39], and the model-free method of StPMKL [46]. The comparison with SVM serves the purpose of a baseline, i.e. to see how far would the gold-standard off-the-shelf method allow one to treat label noise as a normal part of any classification problem. The comparison with rKFD allows us to see comparative performance between generative and discriminative model

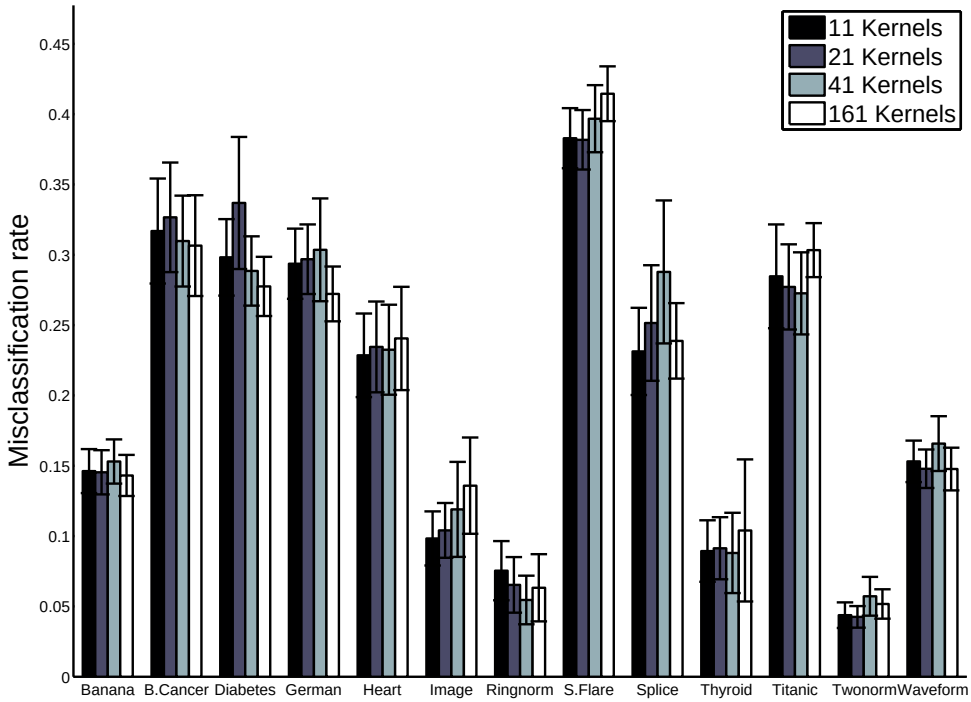

Fig. 5. Comparison of the different kernel set size at 30% level, averaged over 20 random runs.

in noisy settings. The comparison with RP-KernelPerceptron will assess how our method performs against an earlier approach that needs access to the true label-flipping proportions. Finally, the comparison with StPMKL will give us insights into the effects of our simple modelling assumptions about the label flipping process in comparison with the model-free approach in StPMKL. We should mention that there exist label-noise robust versions of SVM [41,2] that could be a subject of further comparisons. However, these works again assume knowledge of label noise probability without providing an algorithm for automatically inferring the noise rate from data, and [41] limit themselves to linear kernels which means no procedure is proposed for the highly non-trivial problem of setting kernel parameters in label-noise conditions. Although for these reasons a direct comparison with these methods would not be particularly meaningful, we believe that our approach for estimating these crucial parameters could be adapted to extend those methods as well in future developments.

3.4.1 rMKLR versus rKFD versus SVM

To make a fair comparison of our rMKLR to its generative counterpart rKFD [24], as well as the 'gold standard' SVM, the kernel widths for rKFD and SVM (and additionally SVM's C parameter) were selected using CV without a trusted validation set for all methods. We perform 100 independent repeated experiments for symmetric and asymmetric noise which ultimately gives us

200 repetitions in total, and we do this at 10% and 30% levels of label noise contamination. Table 5 summarises our findings.

Dataset	10% Noise			
	rKFD	SVM	rMKLR	p-value
Banana	12.39 ± 1.13	11.55 ± 1.07	11.39 ± 0.79	0.35
B.Cancer	28.71 ± 4.81	27.90 ± 5.05	27.93 ± 4.50	0.61
Diabetes	27.15 ± 2.51	24.21 ± 2.07	24.56 ± 2.00	0.12
German	26.93 ± 2.64	24.73 ± 2.57	**24.12 ± 2.38**	1.58e-2
Heart	18.96 ± 4.10	17.60 ± 3.92	17.27 ± 3.48	0.37
Image	5.25 ± 0.88	**4.95 ± 0.85**	6.09 ± 1.19	1.21e-5
Ringnorm	2.32 ± 0.44	**1.84 ± 0.49**	2.20 ± 0.48	5.16e-19
S.Flare	35.37 ± 1.91	**34.25 ± 2.24**	34.93 ± 1.96	1.34e-3
Splice	15.09 ± 1.47	**13.12 ± 1.14**	15.11 ± 1.80	6.54e-8
Thyroid	7.07 ± 3.96	6.03 ± 3.13	6.15 ± 2.75	0.31
Titanic	24.22 ± 2.23	**22.88 ± 1.27**	23.00 ± 1.12	4.04e-3
Twonorm	**2.61 ± 0.29**	2.88 ± 0.52	2.91 ± 0.36	1.67e-20
Waveform	12.82 ± 1.40	11.12 ± 1.06	10.93 ± 0.76	0.27
Dataset	30% Noise			
	rKFD	SVM	rMKLR	p-value
Banana	20.42 ± 6.07	17.63 ± 5.21	**14.92 ± 2.83**	1.35e-10
B.Cancer	33.50 ± 8.21	32.95 ± 8.39	32.36 ± 8.98	0.31
Diabetes	34.47 ± 4.77	29.60 ± 3.94	**26.87 ± 3.75**	2.30e-12
German	32.34 ± 4.56	29.80 ± 4.11	**27.75 ± 4.68**	2.60e-8
Heart	26.49 ± 9.18	25.31 ± 8.21	**23.49 ± 8.69**	1.66e-3
Image	12.41 ± 3.00	10.24 ± 2.33	10.31 ± 3.12	0.75
Ringnorm	6.88 ± 2.33	**3.24 ± 1.95**	4.51 ± 2.43	5.02e-10
S.Flare	38.51 ± 4.12	38.65 ± 4.21	**37.07 ± 3.77**	9.46e-5
Splice	29.89 ± 5.52	**21.46 ± 2.34**	26.98 ± 6.61	1.39e-4
Thyroid	15.93 ± 8.76	12.65 ± 7.99	**8.87 ± 8.89**	1.40e-12
Titanic	29.25 ± 8.86	**27.80 ± 8.36**	28.12 ± 7.41	3.38e-2
Twonorm	**3.08 ± 1.48**	5.38 ± 2.57	4.42 ± 1.77	9.62e-38
Waveform	19.75 ± 3.38	16.40 ± 3.33	**14.15 ± 2.39**	9.54e-15

Table 5
Comparative performance of rKFD and SVM by CV and the proposed rMKLR. Average errors, standard deviations and p-values between the best and the second best performer at 5% level are reported. Boldface entries are statistically superior to the rest.

We observe rMKLR tends to outperform the rKFD at 10% noise but the differnce is not statistically significant on most of the datasets. However it constantly dominates as the noise level increases. It is thus apparent that, as far as classification is concerned, a discriminative classifier such as rMKLR has an edge over rKFD.

The SVM is doing incredibly well as a straight-out-of-the-box classifier, espe-

cially at low noise levels (10%). There is no doubt that slack variables play an important role in SVM's robustness. Nevertheless, rMKLR wins over at higher noise levels (30%). Overall, these results also suggests that explicit label noise treatment is required for traditional classifiers to perform well under label noise.

3.4.2 rMKLR versus RP-KernelPerceptron

Among the very few existing methods for nonlinear classification in label noise conditions, the RP-KernelPerceptron [39] develops an extension of kernel perceptron that is able to deal with label flipping, provided that the flipping probabilities are known beforehand. Having knowledge of the flipping proportions helps reduce the search space in the space of possible classifiers, since it is non-trivial to decide when to trust the training labels and when to trust the classifier outputs on the training points instead. A non-linrear classifier is very flexible and can easily learn the noise and overfit.

Hence, the problem tackled in [39] is easier than the problem tackled in our approach where the flipping probabilities are estimated from the data. Therefore we might expect better performance from a method that has access to the true label flipping proportion than a method that does not. We performed comparisons on the data sets used in [39], against the quoted results from the companion technical report by the same authors [40]. The results are presented in Table 6. Interestingly, as we can see, we found the performances are still comparable despite RP-KernelPerceptron had access to the true flipping probabilities whereas our method has estimated them from the data.

Dataset	Noise level					
	10%			30%		
	rp-KernelP	rMKLR	p-value	rp-KernelP	rMKLR	p-value
Banana	12.73 ± 1.36	**11.55 ± 0.78**	4.01e-27	19.61 ± 3.57	**16.69 ± 2.74**	4.92e-18
B.Cancer	28.34 ± 4.38	28.11 ± 4.57	0.63	31.13 ± 6.46	**29.46 ± 6.34**	0.01
Diabetes	24.25 ± 1.98	24.62 ± 2.03	0.07	**26.69 ± 2.88**	27.85 ± 8.47	2.46e-4
German	24.53 ± 2.54	24.17 ± 2.43	0.15	27.73 ± 3.21	28.00 ± 3.12	0.38
Heart	16.93 ± 4.17	17.35 ± 3.51	0.23	**20.97 ± 5.04**	24.85 ± 11.03	6.38e-4

Table 6
Comparative performance of RP-KernelPerceptron that had access to the true noise rates versus our proposed rMKLR where the noise rates are estimated from the data, on the data sets used in [39,40]. Average errors, standard deviations and p-values at 5% level are reported, as computed using two-tailed t-tests. The results for RP-KernelPerceptron are quoted from [40].

3.4.3 rMKLR versus StPMKL

As a final experiment, before diving into real applications, we compare our rMKLR to the model-free multiple kernel learning algorithm for noisy labels called StPMKL [46]. We follow the experimental protocol discussed in [45,46] to generate multiple RBF kernels with 10 different widths $\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$ for individual features as well as for all features, leading to $S = 10(m + 1)$ kernels in total for each data set – where m is the dimensionality of the data. In addition to the *Heart* data set from the 13 benchmark data sets we analysed before, we use *Ionosphere* and *Australia* data sets from UCI repository in this experiment so that we can compare direcly with quoted results for StPMKL. We perform 20 repetitions using 80% train and 20% test random split. The statistics of the data sets used in this experiment are summarised in Table 7.

Data set	# of Examples	Dimensionality	# of Kernels
Ionosphere	351	34	350
Heart	270	13	140
Australia	690	14	150

Table 7
Statistics of data sets used in the comparison between the MKL algorithms rMKLR (our method) and St-PMKL [46].

Figure 6 shows the classification accuracy of the algorithms with label noise levels varied from 0% to 40% on the three data sets. The results for StPMKL are quoted from [46]. We observe that the performance of rMKLR is similar to that of StPMKL when there is either (i) no label noise or (ii) mild label noise, but rMKLR tends to perform better when the label noise level is high. However, we suspect that the experimental procedure that generates multiple kernels for each feature is not particularly suitable for rMKLR due to the sparsity promoting regularisation that the algorithm used. We then repeat the experiment with multiple kernels generated from full-length input vectors (not each feature individually). The results, denoted as 'rMKLR*', turn out to be very interesting. We observe significant boost in classification accuracy and see that rMKLR* outperforms StPMKL in almost all cases. The results also demonstrate convincingly that a model-based approach does not over-simplify the label noise problem and it is practically useful.

3.5.1 Recognising Textual Entailment

For the first real world problem, we test the proposed method on a variant of the PASCAL2 competition data discussed in [38,36]. The data set contains 800 sentence pairs. An annotator was asked if the second sentence follows from the first sentence. There are 164 distinct annotators in total, of which only one annotator has labelled all sentence pairs. On average an annotator has completed 53 out of 800 pairs which results in a sparse 800×164 matrix. Apart from that, the actual ground truths are also given. The task is to estimate and predict the ground truths using a varying number of annotators. For this type of task, majority voting has long been a standard approach but [36] has already demonstrated that we can do better. This is apparent in our results too. We measure the accuracy of the estimated ground truth while varying the number of annotators, at which point we perform 100 independent random repetitions. The overall results together with those quoted from [36] are summarised in Figure 7.

We find that rMKLR uses less annotations to achieve the same accuracy as the majority voting. We further observe that rMKLR statistically outperforms the EM-algorithm based approach (as employed in [36]) when limited annotations are provided. This is because the model discussed in [36] is formulated such that each annotator has their own 'flip matrix' whereas ours employs a single

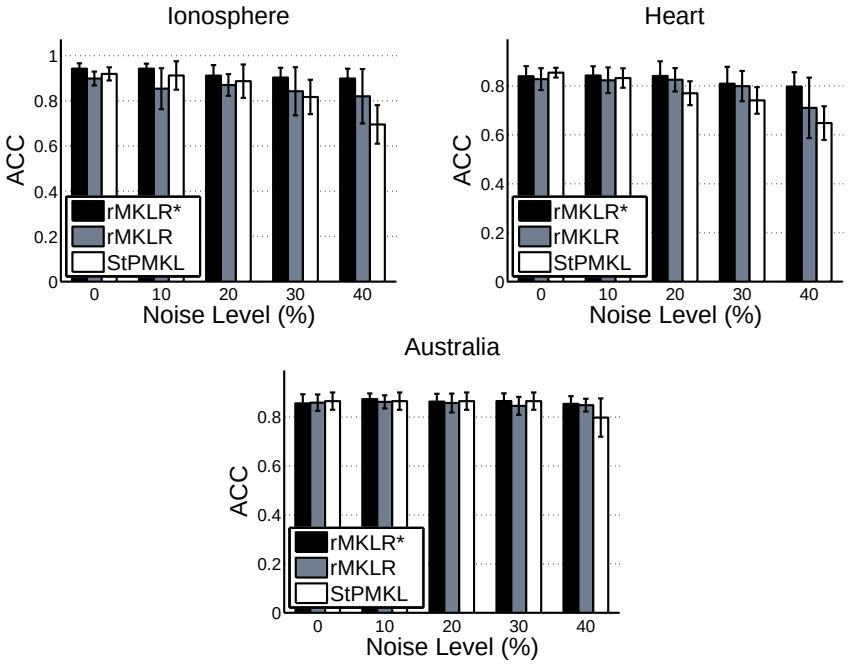

Fig. 6. Comparison of classification accuracy (ACC) with noise level ranging from 0% to 40%.

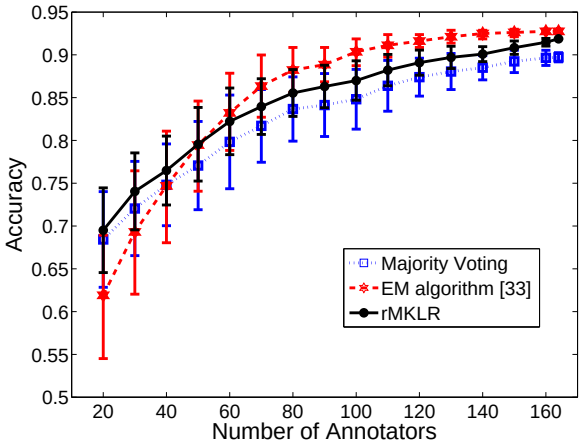

Fig. 7. Accuracy vs number of annotators in Textual Entailment recognition task. Each result is obtained by 100 independent draws without replacement from the total of 164 annotators.

flip matrix. The pro of having separate 'flip matrices' is to be able to assess the quality of each annotator separately, but the algorithm inevitably requires more labels in order to perform. A single flip matrix, however, suffers less from scarcity of labels but has to pay the price when more labels become available.

3.5.2 Image classification using cheaply acquired labelled data

It is well reckoned that careful labelling of large amounts of data by human experts is extremely tiresome. Suppose we were to train a classifier to recognise images that contain a bike. The standard machine learning approach is to collect training images representing 'bike', as well a counterexamples, and laboriously label each of them. Here, we suggest that we could reduce human intervention and obtain the training data cheaply using annotated data from search engines. By searching for images using the keyword 'bike' and 'not bike', we obtain a set of images that are *loosely* categorised into 'bike' class versus 'not bike' class. This allows us to acquire a large number of training data quickly and cheaply. The problem is of course that the annotations returned by the search engine are somewhat unreliable. Here we demonstrate that the proposed model is useful in such circumstances. We collected 515 images using the keyword 'bike' and 515 images using the keyword 'not bike' from Google [3]. We also manually labelled all images, but will not use these labels for training. The manual labels were determined as the following: a 'bike' image is one that contains a bike as its main object and we make no distinction between a bicycle and a motorbike. Everything else is labelled as 'not bike'. This reveals 83 flips from 'bike' to 'not bike' images and 100 flips from 'not bike' to 'bike' relative to the labels from the search engine. The manually labelled set is

[3] Available at: http://cs.bham.ac.uk/~jxb008/data/websearch.zip

only used for testing purposes. Now, the images are passed through a series of pre-processing steps: we extracted a meaningful visual vocabulary using dense SIFT [26], then extracted texture information using Local Binary Pattern (LBP) [33], and finally extracted Pyramid Histogram of Oriented Gradients (PHOG) descriptors [13]. Having three distinct types of features allows us to exploit the original idea behind MKL where heterogeneous data are combined. We construct 21 RBF base kernels for each types of feature, which results in 63 base kernels in total. We employ rMKLR to learn logistic regression parameters as well as the combination of kernels.

We repeated 100 independent bootstrap classification experiments using 80/20 random splits and employed KLR, rMKLR and additionally linear rLR to perform the task comparatively. The rMKLR attains an average generalisation error of 14.19% $\pm$ 0.02 while traditional KLR and linear rLR lag behind. This result is summarised in Table 8, and highlights the advantage of our new robust kernel machine and how badly KLR was affected by label noise. A subset of classification results from rMKLR are depicted in Figure 8 and Figure 9 for visual inspection. We see that rMKLR is able to detect mislabelled instances effectively. On the basis of these results we believe that there is high potential for learning from unreliable data from the Internet using the label-noise robust algorithm proposed.

Classifier	rLR	KLR	rMKLR
Error rate	18.17% $\pm$ 0.02	21.44% $\pm$ 0.03	**14.19% $\pm$ 0.02**

Table 8
Comparative results between rMKLR, KLR and linear rLR on the noisy label image classification task. The proposed rMKLR is the best performer. Interestingly, linear rLR also outperforms the traditional KLR.

4 Discussion and possible extensions

4.1 Extension to multi-class problems

The proposed multi-kernel approach with Bayesian regularisation technique can be straightforwardly extended to a multi-class problem. In multi-class setting the class posterior of the true label is typically modelled by the softmax function

$$p(y = k|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}_k) = \frac{\exp(\mathbf{w}_k^T \kappa(\cdot, \mathbf{x}_n))}{\sum_{j=0}^{K-1} \exp(\mathbf{w}_j^T \kappa(\cdot, \mathbf{x}_n))} \tag{26}$$

Fig. 8. Examples of positive class ('Bike') predictions sorted by their posterior probability. Boxed images illustrate disagreement between the classifier (denoted as P)

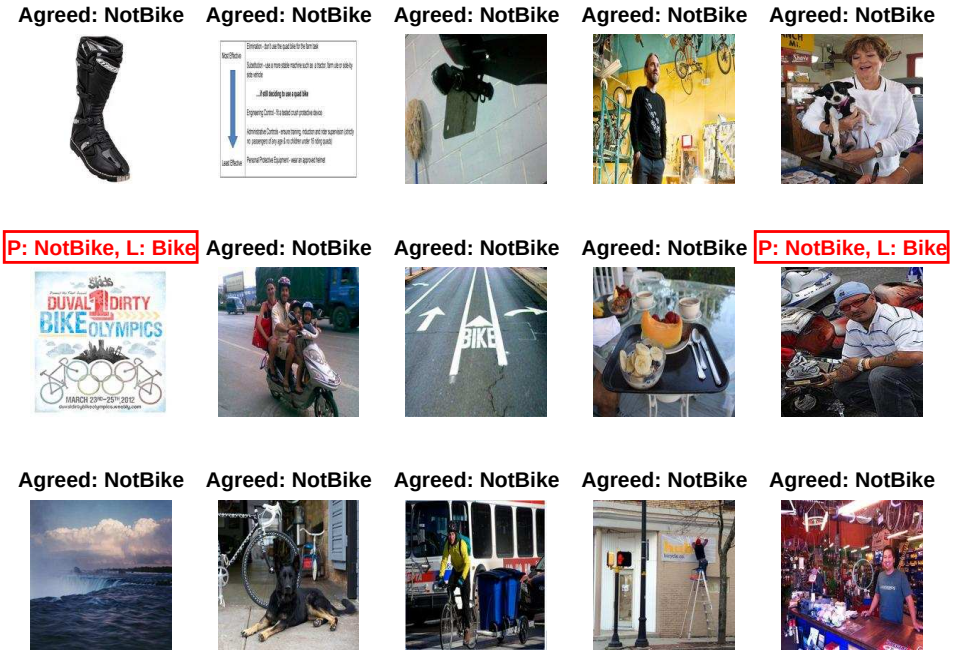

Fig. 9. Examples of negative class ('NotBike') predictions sorted by their posterior probability. Boxed images illustrate disagreement between the classifier (denoted as P) and the provided labels from Google (denoted as L).

Using this we can write the likelihood of the observed label as the following:

$$p(\tilde{y} = k|\kappa(\cdot, \mathbf{x}_n), \Theta) = \sum_{j=0}^{K-1} \omega_{jk} p(y = j|\kappa(\cdot, \mathbf{x}_n), \mathbf{w}_j) \tag{27}$$

which brings us to the objective of the 'robust Multi-Class Multi-Kernel Logistic Regression'.

$$\sum_{n=1}^{N}\sum_{k=1}^{K-1}\mathbb{1}(\tilde{y}_n=k)\log\tilde{P}_n^k-\sum_{k=1}^{K-1}\zeta_k\sum_{n=1}^{N}w_{nk}^2-\sum_{i=1}^{S}\xi_i\eta_i \tag{28}$$

The optimisation of the objective proceeds in the same way as in the binary case by using a conjugate gradient method. The gradient of the objective w.r.t $\mathbf{w}_c$ is given by:

$$\begin{aligned}\mathbf{g}_{\mathbf{w}_c}=&\sum_{n=1}^{N}\sum_{k=0}^{K-1}\frac{\mathbb{1}(\tilde{y}_n=k)}{\tilde{P}_n^k}\frac{\left(\sum_{j=0}^{K-1}(\omega_{ck}-\omega_{jk})e^{(\mathbf{w}_j^T\kappa(\cdot,\mathbf{x}_n))}\right)e^{(\mathbf{w}_c^T\kappa(\cdot,\mathbf{x}_n))}\cdot\kappa(\cdot,\mathbf{x}_n)}{\left(\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\kappa(\cdot,\mathbf{x}_n))}\right)^2}\\&-\zeta_c\sum_{n=1}^{N}w_{nc}^2\end{aligned} \tag{29}$$

And w.r.t u_i:

$$\begin{aligned}\mathbf{g}_{\phi_i}=&\sum_{n=1}^{N}\sum_{c=0}^{K-1}\sum_{k=0}^{K-1}\frac{\mathbb{1}(\tilde{y}_n=k)}{\tilde{P}_n^k}\frac{\left(\sum_{j=0}^{K-1}(\omega_{ck}-\omega_{jk})e^{(\mathbf{w}_j^T\kappa(\cdot,\mathbf{x}_n))}\right)e^{(\mathbf{w}_c^T\kappa(\cdot,\mathbf{x}_n))}(\mathbf{w}_c^T\kappa_i(\cdot,\mathbf{x}_n))}{\left(\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\kappa(\cdot,\mathbf{x}_n))}\right)^2}\\&-2\xi_iu_i\end{aligned} \tag{30}$$

Since we treat weight vectors of each class separately, the regularisation parameters can then be determined using eq.(24) and eq.(15) without the need of modification. Further, the estimates of the elements of the flip matrix ω_{jk} can be obtained by efficient multiplicative update equations:

$$\omega_{jk}=\frac{1}{C}\times\omega_{jk}\sum_{n=1}^{N}\frac{\mathbb{1}(\tilde{y}_n=k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T\mathbf{x}_n)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}} \tag{31}$$

where the constant term C equals $\sum_{k=0}^{K-1}\omega_{jk}\sum_{n=1}^{N}\frac{\mathbb{1}(\tilde{y}_n=k)}{\tilde{P}_n^k}\frac{e^{(\mathbf{w}_j^T\mathbf{x}_n)}}{\sum_{l=0}^{K-1}e^{(\mathbf{w}_l^T\mathbf{x}_n)}}$.

4.2 Limitations and outlook

Throughout this study, we studied random label noise – that is the modelling assumption that mislabelling arises randomly and independently from the contents of the observation features. Examples of real situations where random label noise is encountered include crowdsourcing data where a user might sometime not even look at the observation before giving out the label. It is worth noting that there are also other types of label noise in the literature: The adversarial label noise is when label flipping occurs as a consequence of an adverse change in an observation vector to mislead a classifier as much as

possible (e.g. in a spam filtering task) [2]. Malicious label noise is when label flipping happens more near the decision boundary [42].

The difference in the nature of label noises suggest that each type of noise would need a separate treatment using an appropriate model. Moreover, label noise that is dependent on the contents of certain input features could be approached effectively by building a model to encode the specific dependencies. Application-specific domain knowledge would go a long way in devising such models. Having said that, however, somewhat surprisingly we found that the random noise model works pretty well even in the situation where the randomness assumption does not strictly hold true – for example in our bike classification problem where the labeller (Google's image search engine) might have used textual information around an image to determine the label of the image. Another example is microarray classification [5] where it is very likely that the labelling process is not entirely independent of the feature content, and yet the random noise model has been successfully applied. In such situations that random label noise may be understood as a simplifying model assumption which trades some suboptimality for tractability to estimate the key parameters from limited amounts of data. These results demonstrate indeed that the random label noise assumption is practically useful despite its simplicity even when there exist some dependencies between the label flipping and the input. Nonetheless, developing more specialised noise models is an interesting avenue for future work.

5 Conclusions

We proposed a novel algorithm to learn a label-noise robust Kernel Logistic Regression model in which the optimal hyper-parameters are automatically determined using multiple kernel learning and Bayesian regularisation techniques. The experimental results show that the latent variable model used is robust against mislabelling while the proposed learning algorithm is faster and has superior predictive abilities than traditional approaches. In comparisons with three state-of-the-art kernel machines in controlled settings we observed significant improvements over the previously existing Kernel Fisher Discriminant classifier and even the Multiple Kernel Learning algorithm developed specifically for noisy labels. Finally, we demonstrated real-world applications to learning from crowd-sourcing data, learning from cheaply obtained but unreliable annotated data.

References

[1] R. Barandela, E. Gasca, Decontamination of training samples for supervised pattern recognition methods, in: Advances in Pattern Recognition, vol. 1876 of Lecture Notes in Computer Science, Springer, 2000, pp. 621–630.

[2] B. Biggio, B. Nelson, P. Laskov, Support vector machines under adversarial label noise, Journal of Machine Learning Research - Proceedings Track 20 (2011) 97–112.

[3] J. Bootkrajang, A. Kabán, Multi-class classification in the presence of labelling errors, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2011, 2011, pp. 345–350.

[4] J. Bootkrajang, A. Kabán, Label-noise robust logistic regression and its applications, in: ECML/PKDD (1), 2012, pp. 143–158.

[5] J. Bootkrajang, A. Kabán, Classification of mislabelled microarrays using robust sparse logistic regression, Bioinformatics 29 (7) (2013) 870–877.

[6] C. Bouveyron, S. Girard, Robust supervised classification with mixture models: Learning from data with uncertain labels, Pattern Recognition 42 (11) (2009) 2649–2658.

[7] C. E. Brodley, M. A. Friedl, Identifying mislabeled training data, Journal of Artificial Intelligence Research 11 (1999) 131–167.

[8] G. C. Cawley, N. L. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, Journal of Machine Learning Research 99 (2010) 2079–2107.

[9] G. C. Cawley, N. L. C. Talbot, Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters, J. Mach. Learn. Res. 8 (2007) 841–861.

[10] G. Celeux, S. Chrtien, F. Forbes, A. Mkhadri, A component-wise em algorithm for mixtures, Journal of Computational and Graphical Statistics 10 (4) (2001) pp. 697–712.

[11] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27.

[12] R. S. Chhikara, J. McKeon, Linear discriminant analysis with misallocation in training samples, Journal of the American Statistical Association 79 (388) (1984) 899–906.

[13] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, 2005, pp. 886 –893.

[14] T. Damoulas, M. A. Girolami, Pattern recognition with a bayesian kernel combination machine, Pattern Recognition Letters 30 (1) (2009) 46 – 54.

[15] B. Frénay, G. de Lannoy, M. Verleysen, Label noise-tolerant hidden markov models for segmentation: Application to ECGs, in: ECML/PKDD (1), 2011, pp. 455–470.

[16] M. Gönen, E. Alpaydin, Multiple kernel learning algorithms, J. Mach. Learn. Res. 12 (2011) 2211–2268.

[17] J. A. Hausman, J. Abrevaya, F. M. Scott-Morton, Misclassification of the dependent variable in a discrete-response setting, Journal of Econometrics 87 (2) (1998) 239–269.

[18] Y. Jiang, Z.-H. Zhou, Editing training data for knn classifiers with neural network ensemble, in: Advances in Neural Networks, vol. 3173 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004, pp. 356–361.

[19] G. S. Kimeldorf, G. Wahba, Some results on Tchebycheffian spline functions, Journal of Mathematical Analysis and Applications 33 (1) (1971) 82–95.

[20] T. Krishnan, S. C. Nandy, Efficiency of discriminant analysis when initial samples are classified stochastically, Pattern Recognition 23 (5) (1990) 529–537.

[21] P. A. Lachenbruch, Discriminant analysis when the initial samples are misclassified, Technometrics 8 (4) (1966) 657–662.

[22] P. A. Lachenbruch, Discriminant analysis when the initial samples are misclassified ii: Non-random misclassification models, Technometrics 16 (3) (1974) pp. 419–424.

[23] G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, W. S. Noble, A statistical framework for genomic data fusion, Bioinformatics 20 (16) (2004) 2626–2635.

[24] N. D. Lawrence, B. Schölkopf, Estimating a kernel fisher discriminant in the presence of label noise, in: Proceedings of the 18th International Conference on Machine Learning, Morgan Kaufmann, 2001, pp. 306–313.

[25] Y. Li, L. F. Wessels, D. de Ridder, M. J. Reinders, Classification in the presence of class noise using a probabilistic kernel fisher method, Pattern Recognition 40 (12) (2007) 3349–3357.

[26] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, ICCV '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 1150–1157.

[27] G. Lugosi, Learning with an unreliable teacher, Pattern Recogn. 25 (1992) 79–87.

[28] L. S. Magder, J. P. Hughes, Logistic regression when the outcome is measured with uncertainty, American Journal of Epidemiology 146 (2) (1997) 195–203.

[29] J. I. Maletic, A. Marcus, Data cleansing: Beyond integrity analysis, in: Proceedings of the Conference on Information Quality, 2000, pp. 200–209.

[30] A. Malossini, E. Blanzieri, R. T. Ng, Detecting potential labeling errors in microarrays by data perturbation, Bioinformatics 22 (17) (2006) 2114–2121.

[31] F. Muhlenbach, S. Lallich, D. A. Zighed, Identifying and handling mislabelled instances, Journal of Intelligent Information Systems 22 (1) (2004) 89–109.

[32] S. W. Norton, H. Hirsh, Classifier learning from noisy data as probabilistic evidence combination, in: Proceeding of the 10th National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 141–146.

[33] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (7) (2002) 971 –987.

[34] P. Pavlidis, J. Weston, J. Cai, W. N. Grundy, Gene functional classification from heterogeneous data, in: Proceedings of the fifth annual international conference on Computational biology, RECOMB '01, ACM, New York, NY, USA, 2001, pp. 249–255.

[35] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for adaboost., Machine Learning (2001) 287–320.

[36] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, L. Moy, Learning from crowds, Journal of Machine Learning Research 11 (2010) 1297–1322.

[37] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (7) (2003) 1015–1022.

[38] R. Snow, B. O'Connor, D. Jurafsky, A. Y. Ng, Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks., in: EMNLP, 2008, pp. 254–263.

[39] G. Stempfel, L. Ralaivola, Learning kernel perceptrons on noisy data and random projections, in: Algorithmic Learning Theory (ALT), Lecture Notes in Computer Science, vol. 4754.

[40] G. Stempfel, L. Ralaivola, Learning kernel perceptrons on noisy data and random projections (2007).
URL `http://hal.archives-ouvertes.fr/hal-00137941`

[41] G. Stempfel, L. Ralaivola, Learning svms from sloppily labeled data., in: ICANN (1), vol. 5768 of Lecture Notes in Computer Science, Springer, 2009, pp. 884–893.

[42] T. Takenouchi, S. Eguchi, N. Murata, T. Kanamori, Robust boosting algorithm against mislabeling in multiclass problems, Neural Computation 20 (6) (2008) 1596–1630.

[43] H. Xu, C. Caramanis, S. Mannor, Robustness and regularization of support vector machines, Journal of Machine Learning Research 10 (2009) 1485–1510.

[44] L. Xu, K. Crammer, D. Schuurmans, Robust support vector machine training via convex outlier ablation, in: Proceedings of the 21st national conference on Artificial intelligence - Volume 1, AAAI'06, 2006, pp. 536–542.

[45] Z. Xu, R. Jin, H. Yang, I. King, M. R. Lyu, Simple and efficient multiple kernel learning by group lasso, in: ICML, 2010, pp. 1175–1182.

[46] T. Yang, M. Mahdavi, R. Jin, L. Zhang, Y. Zhou, Multiple kernel learning from noisy labels by stochastic programming, in: ICML, 2012, pp. 233–240.

[47] Y. Yasui, M. Pepe, L. Hsu, B.-L. Adam, Z. Feng, Partially supervised learning using an em-boosting algorithm, Biometrics 60 (1) (2004) 199–206.

A Notations and symbols used in the paper

Symbol	Description	Symbol	Description
$\mathcal{D}$	A dataset	$\mathbf{x}$	A data point
y	True label	$\tilde{y}$	Observed label
$\hat{y}$	Predicted label	N	Number of data points
m	Dimensionality of data	K	Number of classes
κ	A kernel	S	Number of kernels
$\mathbf{w}$	Logistic regression parameter vector	ζ	Regularisation on $\mathbf{w}$
$\boldsymbol{\eta}$	Kernel combination coefficient vector	ξ	Regularisation on $\boldsymbol{\eta}$
Ω	Label flipping probability matrix	ω_{jk}	Element of Ω

B List of abbreviations

Abbreviation	Description
LR	Logistic Regression
rLR	robust Logistic Regression
KLR	Kernel Logistic Regression
rKLR	robust Kernel Logistic Regression
rMKLR	robust Multiple Kernel Logistic Regression
rKFD	robust Kernel Fisher Discriminant
StPMKL	Multiple Kernel Learning by Stochastic Programming